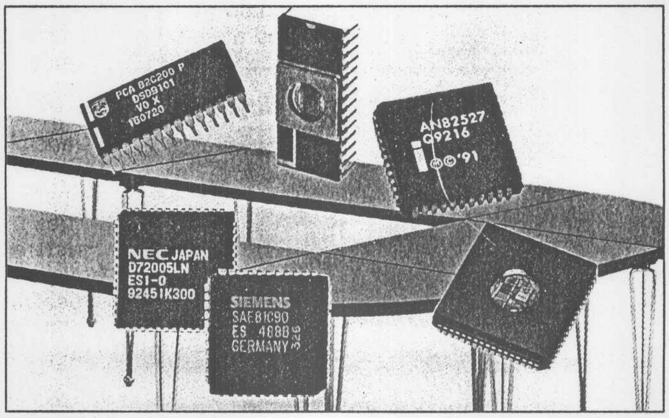
An Inside Look at the Fundamentals of CAN

HOLGER ZELTWANGER, CAN in Automation Users and Manufacturers Group e.V., Erlangen, Germany

Originally designed for automobiles, the Controller Area Network (CAN) is a serial bus with large potential use in industry.



Bosch has granted CAN licenses to a number of semiconductor manufacturers; stand-alone CAN controllers are currently available from NEC, Intel, Philips, and Siemens. Microcontrollers with integrated CAN controllers—so-called single chip solutions—are produced by Intel, Philips, Siemens, Motorola, and National Semiconductor. IAM and Inicore produce gate arrays with CAN controllers.

Electronic control devices on automobiles have been in use for some time. They control the transmission, engine timing, and fuel injection, as well as sophisticated systems like acceleration skid control (ASC) and antilock brake systems.

The complexity of these functions requires an exchange of data. As they become more complex, hard-wired signal lines become cumbersome and expensive. In the case of advanced automotive control systems, such as Bosch's Motronic, the number of connections cannot be increased much further.

Moreover, a number of systems implement functions covering more than one control device. ASC requires the interplay of engine timing and fuel injection to reduce torque when drive wheel slippage occurs. With electronic transmission control, ease of gear changing can be improved by a brief adjustment to ignition timing.

The limitations of conventional control device linkage can be overcome by networking the system components using a serial data bus system. It was for this reason that Bosch developed the Controller Area Network (CAN), which has since been standardized internationally (ISO 11898) and has been cast in silicon by several semiconductor manufacturers.

Using CAN, peer stations (controllers, sensors, and actuators) are connected via a serial bus. The protocol corresponds to the data link layer (layer 2) in the ISO/OSI reference model. Unlike cable trees, the network protocol detects and corrects communication errors caused by electromagnetic interference. The network is relatively easy to configure, and allows any station to communicate with any other station without putting too great a load on the central control computer.

The spread of CAN

There are great similarities in the requirements for motor vehicle bus systems and industrial fieldbus systems. They both need to be low cost and they need to operate in harsh electrical environments. High real-time capabilities and ease of use are equally desirable.

The standard use of CAN in Mercedes-Benz's S Class cars and the adoption of CAN by U.S. commercial vehicle manufacturers has not escaped the notice of industrial users. Manufacturers of agricultural and marine equipment have chosen to use CAN; it is also the choice of some manufacturers of medical apparatus, textile machines, special-purpose machinery, and elevator controls. The serial bus system is particularly well-suited to networking intelligent I/O devices as well as sensors and actuators within a machine or plant.

The textile machinery industry is one of the pioneers of CAN. As early as 1990, one manufacturer equipped its looms with modular control systems communicating in real time via CAN networks. Several textile machinery manufacturers have joined together to form the CAN Textile Users Group, which in turn is a member of the international users and manufacturers group CAN in Automation (CiA).

In the U.S. a number of enterprises are using CAN in production lines and machine tools as an internal bus system for networking sensors and actuators within the line or machine. Some users in the medical engineering sector decided in favor of CAN because they had particularly stringent safety requirements. Other manufacturers of machinery and equipment, such as robots and transport systems, face similar safety requirements.

Apart from the high communication reliability, the low connection costs per station are a further decisive argument for CAN. In applications where price is critical it is essential that CAN chips be available from a variety of manufacturers. In some applications, such as low voltage switchgear, the compactness of the controller chip is also important.

How CAN functions

When data are transmitted by a CAN node, they are broadcast to all other stations. A receiving station may realize the message is not relevant to its purpose and eventually discard it but initially, all stations receive the same message.

The type of data transmitted—engine rpm, oil temperature—is designated by an 11-bit identifier at the start of the message. Most importantly, the identifier also defines the priority of the message. This type of message service is called a "content-oriented addressing scheme."

Unique to CAN is the 'content-oriented addressing scheme'

Each 11-bit identifier is unique in the network. No two nodes can have messages with the same identifier. Likewise, a single node cannot have two different types of messages with the same 11-bit identifier. This is important for bus allocation when several stations are competing for bus access.

If the CPU of a given station wishes to send a message to one or more stations, it passes the data and its identifier to its assigned CAN chip. This is the Make Ready state in Figure 1. The message is constructed and transmitted by the CAN chip when it receives the bus allocation, indicated by the Send Message state.

When this happens, all other stations become receivers of the message (Receive Message). Each receiver performs an acceptance test to determine whether the data are relevant (Select). If so, the data are accepted; otherwise they are ignored.

A high degree of system and configuration flexibility is achieved as a result of the content-oriented addressing scheme. It is easy to add stations to a CAN network without making hardware or software modifications, provided the new stations are purely receivers. The data transmission protocol does not require physical destination addresses for the individual components. It permits synchronization of distributed processes: measurements needed as information by several controllers can be transmitted via the network. This makes it unnecessary for each controller to have its own sensor.

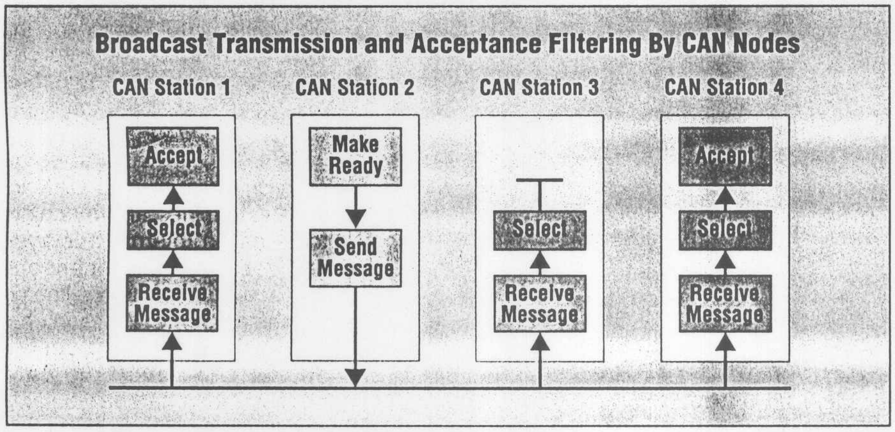


Figure 1: In this example, station 2 broadcasts its message to all the nodes on the network, but only stations 1 and 4 accept the data. Station 3 receives the message, but ignores it.

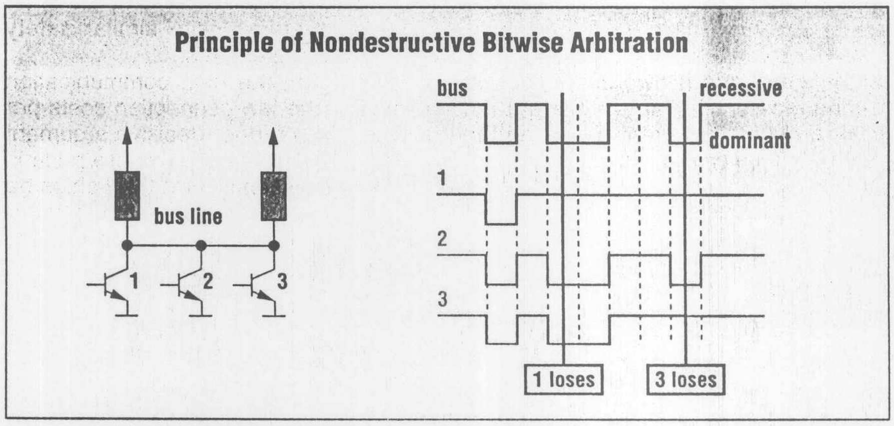


Figure 2: In this example three CAN nodes wish to transmit at the same time. The identifier for node 1 is 011111...; node 2 is 0100110...; node 3 is 0100111... All have the same first two digits, so nothing happens until the third digit is compared and node 1 "loses" because it has a 1 while the other two have 0. Zero is the dominant bit and always "wins" over a one bit, which is recessive. Bit positions 4, 5, and 6 are the same for nodes 2 and 3, but at the seventh bit position node 3 loses because it has a 1 and node 2 has a zero. Note the signal on the bus continually tracks the "winner," which in this case is node 2. The advantage of nondestructive bitwise arbitration is that while the network is in the process of determining which node will be the winner, it is already transmitting the first part of the message. All "losers" automatically become receivers of the message with the highest priority. They do not reattempt transmission until the bus is available again.

Bitwise arbitration

For the data to be processed in real time they must be transmitted rapidly. This requires a physical data transfer path at high speed but it also calls for rapid bus allocation when several stations wish to send messages simultaneously.

In real-time processing the urgency of messages to be exchanged over the network can differ greatly. A rapidly changing variable, such as engine load, has to be transmitted more frequently and therefore with less delay than other parameters, such as engine temperature, which change relatively slowly.

The priority at which a message is transmitted is incorporated in the 11-bit identifier. The identifier with the lowest binary number has the highest priority. These priorities are specified during system design and cannot be changed dynamically. Bus access conflicts are resolved by bitwise arbitration on the identifiers involved by each station. For an example of how this process works, see Figure 2.

CAN is highly efficient because the bus is utilized only by those stations with pending transmission requests. These requests are handled in the order of the importance of the messages for the system as a whole. This proves especially advantageous in heavily loaded situations. Since bus access is prioritized on the basis of the messages, it is possible to guarantee low individual latency times in real-time systems.

To circumvent the problem of the reliability of a master station (and thus of the whole communication system), the CAN protocol implements decentralized bus control. All major communication mechanisms, including bus access control, are implemented several times in the system. This is the only way to fulfill the high requirements for the availability of the communication system.

CAN vs. other schemes

There are basically two important methods of bus allocation used in general practice: allocation on a fixed time schedule and allocation on the basis of need. In the first, allocation is made sequentially to each node for a maximum duration regardless of whether the participant needs the bus or not (example: token passing). With methods of this type the bus is allocated to one and only one station either immediately or

The collapse of a CAN system due to message overload is not possible

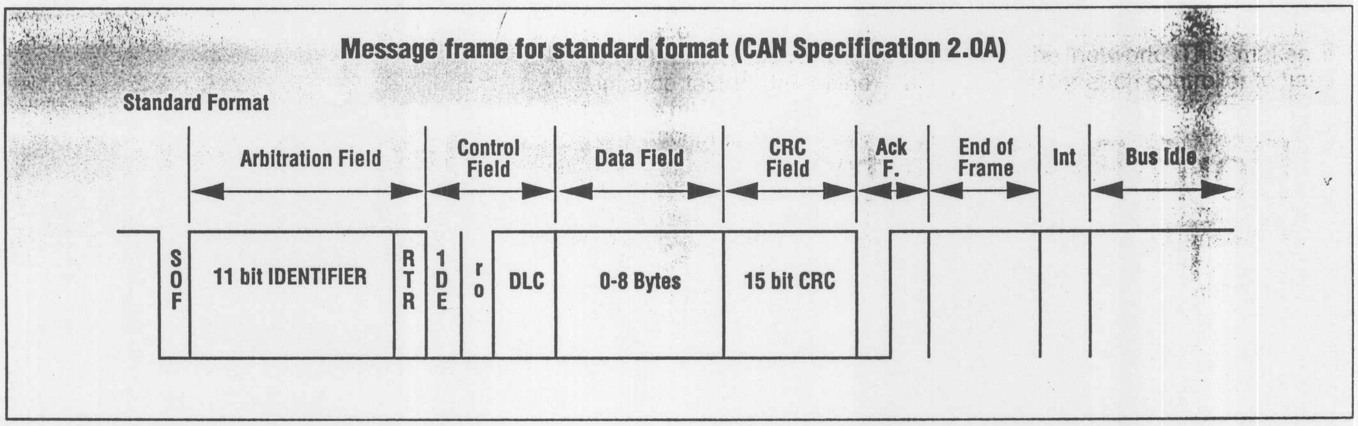


Figure 3: This is the message frame for standard CAN format. The only difference between standard and extended format (CAN specification 2.0B) is an additional 18 bits added to the arbitration field. The bit representation used in NRZ (non-return to zero).

within a specified time following a single bus access (by one or more stations). This ensures that each bus access by one or more stations leads to an unambiguous bus allocation.

In the second general method, the bus is allocated to one participant on the basis of transmission requests outstanding, *i.e.,* the allocation system only considers participants wishing to transmit (example: Ethernet CSMA/CD). In this method, simultaneous bus access by more than one station causes all transmission attempts to be aborted and therefore there is no successful bus allocation. More than one bus access may be necessary in order to allocate the bus at all.

CAN implements a bus allocation method that guarantees unambiguous bus allocation even when there are simultaneous bus access requests from different stations. The method of bitwise arbitration uniquely resolves any collision between stations wanting to transmit, and it does this within a maximum of 13 (standard format) or 33 (extended format) bit periods.

Unlike the message-wise arbitration of Ethernet (CSMA/CD), CAN's nondestructive method of conflict resolution ensures that no bus capacity is used without transmitting useful information.

Even in situations where the bus is heavily loaded the linkage of the bus access priority to the content of the message proves to be a beneficial system attribute. In spite of the insufficient bus transport capacity, all outstanding transmission requests are processed in order of their importance to the overall system (as determined by the message priority). Collapse of the whole system due to overload, as can occur with CSMA/CD networks such as Ethernet, is not possible with CAN.

Message frame formats

The message frame for transmitting messages on the bus is comprised of seven main fields (refer to Figure 3). The CAN protocol supports two message frame formats. The only difference is the length of the identifier (ID). In the standard format the length is 11 bits and in the extended format, 29 bits.

A message in the standard format begins with the start bit called "start of frame" (SOF). This is followed by the arbitration field which contains the 11-bit identifier and the remote transmission request (RTR) bit. The RTR bit indicates whether it is a data frame or a request frame. A request frame does not have data bytes.

The control field contains the identifier extension (IDE) bit, which indicates standard format or extended format. It also has a bit reserved for future extensions (ro) and, in the last four bits, a count of the data bytes in the data field (DLC). The data field ranges from zero to eight bytes and is followed by the cyclic redundancy check (CRC) used as a frame security check for detecting bit errors.

In the U.S., Honeywell, Cutler-Hammer, and Allen-Bradley have developed sensor and actuator networks based on CAN.

The acknowledgement (ACK) field comprises the ACK slot (one bit) and the ACK delimiter. The bit in the ACK slot is put on the bus by the transmitter as a recessive (logical 1) bit. It is overwritten as a dominant bit (logical 0) by those receivers which have at this time received the data correctly. In this way the transmitting node can be assured at least one receiver has correctly received its message. Note that messages are acknowledged by the receivers regardless of the result of the acceptance test.

The end of the message is indicated by the end of frame. The intermission is the minimum number of bit periods separating consecutive messages. If there is no following bus access by any station, the bus remains idle.

Detecting errors

Unlike other bus systems, the CAN protocol does not use acknowledgement messages. Instead, it signals any errors that occur. The CAN protocol implements five error checking mechanisms. The first three are at the message level:

Cyclic Redundancy Check—The CRC safeguards the information in the frame by adding redundant check bits at the transmission end. At the receiver end these bits are recomputed and tested against the received bits. If they do not agree there has been a CRC error.

Frame check—This mechanism verifies the structure of the transmitted frame by checking the bit fields against the fixed format and the frame size. Errors detected by frame checks are designated as format errors.

ACK errors—As mentioned previously, frames received are acknowledged by all recipients through positive acknowledgement. If no acknowledgement is received by the transmitter this may indicate an error detected only by the

CAN implements five error checking mechanisms

recipients, that the ACK field has been corrupted, or there are no receivers.

The CAN protocol also implements two mechanisms for error detection at the bit level:

Bus monitoring—A somewhat unique capability of CAN is that a node can monitor its own signal on the bus while transmitting. Thus, each transmitting node observes the bus level and detects differences between the bit sent and the bit received. This permits reliable detection of global errors and errors local to the transmitter.

Bit stuffing—The bit representation used by CAN is NRZ (non-return-to-zero), which guarantees maximum efficiency in bit coding. However, if there are too many bits in a row with the same value, synchronization can be lost.

To guard against this loss, synchronization edges are generated by means of bit stuffing. After five consecutive equal bits the sender inserts into the bit stream a stuff bit with the complementary value. The stuff bit is automatically removed by the receivers. In other words, after five consecutive zero bits, CAN will automatically insert a one bit. The code check is limited to checking adherence to the stuffing rule. If six bits in a row have equal value, CAN knows there is an error.

If one or more errors are discovered by at least one station using the above mechanisms, it sends an error flag that aborts the current transmission. This prevents other stations from accepting the message and thus ensures the consistency of data throughout the network.

> A CAN network in an automobile is predicted to have one undetected communication error in a thousand years of operation.

After transmission of an erroneous message has been aborted, the sender automatically reattempts transmission. As a rule, retransmission begins within 23 bit periods after error detection. In special cases the system recovery time is 31 bit periods.

However effective and efficient the method described may be, a defective station might lead to all messages (including correct ones) being aborted, thus blocking the bus system if no measures for self-monitoring were taken.

The CAN protocol therefore provides a mechanism for distinguishing sporadic errors from permanent errors and localizing station failures. This is done by statistical assessment of station error situations with the aim of recognizing a station's own defects and possibly entering an operating mode where the rest of the CAN network is not negatively affected. This may go as far as the station switching itself off to prevent messages erroneously recognized as incorrect from being aborted.

Data reliability of CAN

The introduction of safety-related systems in automobiles brought with it requirements for the high reliability of data transmission. The objective is to prevent situations that may endanger the driver as a result of data exchange throughout the whole life of a vehicle.

This goal is achieved if the reliability of the data is sufficiently high or the residual error probability is sufficiently low. In the context of bus systems data, reliability is understood as the capability to identify data corrupted by transmission faults.

The *residual error probability* is a statistical measure of the impairment of data reliability. It specifies the probability that data will be corrupted and that the corruption will remain undetected. Residual error probability must be so small that on average no corrupted data will go undetected throughout the whole life of a system.

The calculation of the residual error probability requires that errors be classified and the whole transmission path be described by a model. If we determine the residual error probability of CAN as a function of the bit error probability for message lengths of 80 to 90 bits, for system configurations of, for instance, five or ten nodes and with an error rate of $^1/_{1000}$ (one error for every thousand messages), then maximum bit error probability is on the order of 10^{-13}.

For example, if a CAN network operates at a data rate of 1 Mbit/sec at an average bus capacity utilization of 50%, for a total operating life of 4,000 hours and with an average message length of 80 bits, then the total number of messages transmitted is 9×10^{10}. The statistical number of undetected transmission errors during the operating life is thus in the order of less than 10^{-2}.

To put it another way, with an operating time of eight hours per day on 365 days per year and an error rate of 0.7 per second, one undetected error occurs

every thousand years, on a statistical average.

Standardization of CAN

The CAN protocol is standardized by ISO 11898. In the U.S., the Society of Automotive Engineers, (SAE) subcommittee on "Truck and Bus" has adopted

In the near future we will see more CAN combined with microcontrollers. CAN will be used in washing machines as well as audio and video devices

the CAN protocol as the basis for further standardization activities (J1939). The ISO is working on a CAN-based protocol for agriculture and forestry machines (ISO WD 11783) and a CAN connection between trucks and trailers (ISO WD11992). There is also some standardization activity in the wheelchair industry to specify a special CAN solution (M3S).

The CAN in Automation organization has developed a protocol corresponding to the application layer in the ISO/OSI reference model. This CAN Application Layer (CAL) specification is license-free. There are five implementors (ESD, I+ME, Janz, Port, and STZP) offering CAL software packages for different microcontrollers and CAN controller chips. Companies who have developed CAL applications include Barmag, Bosch, Moog, Philips Medical Systems, Philips CFT, Selectron, Sauer-Sundstrand, and Weidmuller.

CiA is now working on communication and device profiles based on CAN and CAL. Work drafts are available for low-level I/O, drives, and low-voltage gear devices. There will also be a CAL profile for automotive applications such as garbage trucks, forklift trucks, or road construction machines.

In the U.S., several companies have developed their own sensor and actuator networks based on the CAN protocol. These include Smart Distributed Systems (SDS) from Honeywell Micro Switch, DeviceNet from Allen-Bradley, and Cutler Hammer Control.

CAN silicon

Communication is identical for all implementations of the CAN protocol. There are differences, however, in the extent to which the implementation takes over message transmission from the microcontrollers which follow it in the circuit.

CAN controllers with an intermediate buffer (formerly called BasicCAN chips) have implemented as hardware the logic necessary to create and verify the bitstream according to protocol. They may place a strain on the microcontroller with acceptance filtering, but they require only a small chip area and can therefore be produced at lower cost. In principle they can accept all objects in a CAN network.

CAN objects consist mainly of three components: identifier, data-length code, and actual useful data. CAN controllers with object storage (formerly called FullCAN) function like CAN controllers with intermediate buffers, but also administer certain functions. Where there are several simultaneous requests they determine, for example, which object is to be transmitted first. They also carry out acceptance filtering for incoming objects.

CAN controllers with object storage are designed to take as much strain as possible off the local microcontroller. These CAN controllers require a greater chip area, however, and are therefore more expensive. In addition, they can only administer a limited number of chips.

CAN controllers which combine both principles of implementation are now available. They have object storage, at least one of which is designed as an intermediate buffer. For this reason there is no longer any point in differentiating between BasicCAN and FullCAN.

There are also CAN chips which do not require a following microcontroller. These CAN chips are called SLIO (serial link I/O). They are CAN slaves and have to be administered by a CAN master.

In the near future we will see more and more CAN combined with microcontrollers. The high volume of applications, especially in the automobile industry, guarantees that prices will dramatically decrease. CAN will be used in washing machines as well as audio and video devices. This wide range of applications will make CAN one of the standard serial interfaces integrated in microcontrollers. □

Readers may contact the author at CAN in Automation Users and Manufacturers Group e.V., Am Weichselgarten 26, D-91058 Erlangen, Germany. Tel: +49 9131 601091; Fax: +49 9131 601092 or ***Circle 260*** *on the reader service card.*